Analyzing Huge Amounts of Data for IPF Applications:

Methodology and Results

Allan Knies

Intel Corporation
Advanced Development Team, Intel Compiler Lab

Overview

Goal: Semi-automate the process of analyzing application behavior on candidate IPF processors

- Analyze data for applications that don't have full-time analysis teams
- Enable first level analysis across many applications and microarchitectures
- Understand sensitivity to various microarchitectural features
- Find critical breaking points specific to IPF in cache size, FSB, etc.
- Expose uarch factor interplay on a per application basis (and potentially across compiler optimizations)

Examples mostly from production HPC codes + SPEC

Problem with Goal

It is easier to generate than to analyze…

For important benchmarks such as TPC or SPEC entire teams spend weeks (or more) analyzing a single configuration

IPF has a broad spectrum of HPC and enterprise applications, but limited design resources

Outline

Uarch Factors and Applications

Methodology and Limitations

Analysis of Critical Breaking Points (knees)

Analysis of Critical Factors and Interactions

Uarch Factors and Applications

Which factors were studied?

- Memory bandwidth (GB/sec)
- Memory latency (in ns)
- LLC size ("last level cache" -- 3rd in this case) (in MB)
- LLC (3rd) latency (in cycles)

- Clock speed NOT necessary

Uarch Factors and Applications

IPF HPC Applications (plus others)

- SpecINT, SpecFP
- Kernels (Linpack, Matrix Multiply, GUPs)
- Structural Analysis (linear, nonlinear) (3 apps, 5 workloads)
- Finite Element Analysis (1)
- Crash Analysis (2)
- DFT: Density Function Theory (2)
- MD: Molecular Dynamics (1)
- CFD (3 apps, 4 workloads)

Outline

Uarch Factors and Applications

Methodology and Limitations

Analysis of Critical Breaking Points (knees)

Analysis of Critical Factors and Interactions

Methodology and Limitations

To be practical across many applications and many uarchs, every phase needs to be at least semi-automatic

1. Trace – PIN and pingtrace
2. Simulate – IPF production simulator
3. Script/Analyze/Graph – Perl, JMP, Excel
4. Present -- PPT

Methodology and ***Limitations***

Highly accurate simulators not always reliable for this
- High level parameters cannot be scaled independently without problems (e.g., FSB bandwidth vs outstanding misses)
- Cache associativity not fixed for different cache sizes (12 – 20)
- Memory system model is mostly simple queuing and many parameters are fixed
- Might be better to use a less detailed uarch model

Correlation of traces to full apps has had limited validation

Compiler impact on resource usage very high for IPF

Methodology and ***Limitations***

Cold-cache impact (cold – hot analysis shows between 0% and 20% potential variation)

Few workloads modeled for each application (e.g., like using only TPC to measure database performance)

Lots of curve fitting to data to allow analysis – some data fits well, some does not, some the tool cannot fit

Looking for trends/breaking points – don't read too much into specific numbers

Outline

Uarch Factors and Applications

Methodology and Limitations

Analysis of Critical Breaking Points

Analysis of Factors and Interactions

Analysis of IPF Critical Points

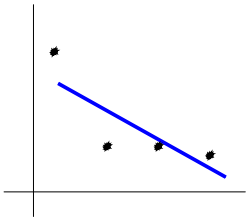
Process
- Simulate 4 points for each uarch factor in question
- Choose points across a range of values centered around likely designs
- Fit a curve to these four points
- Analyze knee of curve to see best cost/perf tradeoff

- Sounds easy -- what if you have 100's of applications and 10's of designs?
- Curve must maintains monotonicity across configurations to automate process

Analysis of IPF Critical Points

Choosing a curve…

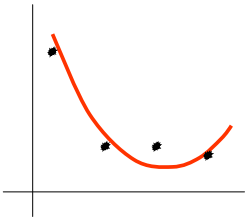
– Linear approximations are monotonic and intuitive, but have high error



Analysis of IPF Critical Points

Choosing a curve …

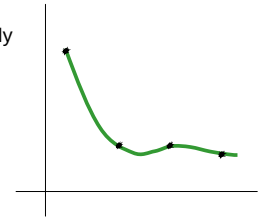
– Using a higher order polynomial yields better RMS error, but are not as useful due to non-monotonicity



Analysis of IPF Critical Points

Choosing a curve …

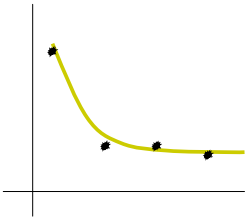
– Neural nets give extremely small RMS error, but are not always monotonic



Analysis of IPF Critical Points

Choosing a curve …

– Exponential decay provides good RMS error and monotonicity



Analysis of IPF Critical Points

Exponential decay formula used:

CPI%(L3Size-MB)
= *theta1* * Exp(-*theta2* * (L3Size-MB) + *theta3*)

JMP chooses theta* parameters to minimize RMS error over sample points

Analysis of IPF Critical Points

Issues

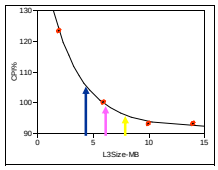
– Curve fitting for a fixed function does not always converge to a solution using JMP

– Some apps insensitive to certain factors
- may never have a slope as small/large as ½ or 2
  – shown as zero values in graphs
- knees are not always detected

Analysis of IPF Critical Points

Example fit from JMP for LLC size on MD code

= Measured pt

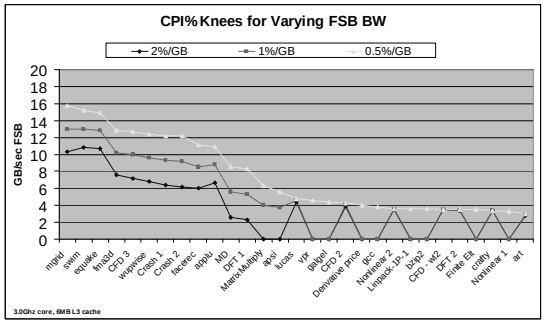


- Take first derivative and solve at various points (-2, -1, and -0.5)
- Give pt where CPI delta is 2%, 1% or ½% for a given LLC size
- Output: "IncreasingSlope, 6.9, 8.9, 10.9"
- ***Spread on X-axis indicates degree of bend in knee***

Analysis of IPF Critical Points

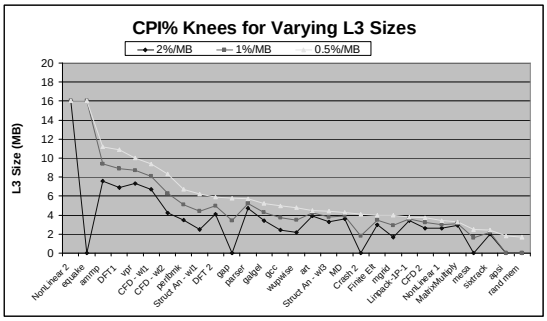
– ***But how do you look at this data for 100 applications?***

Analysis of Critical Points (FSB)



Distance between curves indicates bend in knee – Height of lines shows raw demand

Analysis of Critical Points (L3 Size)



Distance between curves indicates bend in knee – Height of lines shows raw demand

Analysis of IPF Critical Points

IPF Conclusions:

– For about 2/3's of the applications, there is a substantial drop-off at about 2MB/LLC per core/thread.
  - This is likely the minimum size needed (or used) by the compiler to stage data into the middle cache
  - For the other 1/3, the ~7MB is a good minimum size

– 60% of the applications were happy with small FSB bandwidths (0.5 bytes/flop),
  - The top 20% wanted 10+ GB/sec (1-2 byte/flop)

Outline

Uarch Factors and Applications

Methodology and Limitations

Analysis of Critical Breaking Points (knees)

Analysis Factors and Interactions

Factors and Interactions

What percent of applications are affected significantly by changes to uarch factors ("scaled estimates" by factor)

Show CPI impact of various uarch features on application performance

Show per-unit rates of CPI change for given uarch features and general function of factor to CPI

Scaled Estimates by Factor

In 4-dimensional space of factors, fit a plane:

$$CPI = aX + bY + cZ + dW \quad \text{(linear terms)}$$
$$+ eX^2 + fY^2 + gZ^2 + hW^2 \quad \text{(quadratic terms)}$$
$$+ iXY + jXW + kXZ + mYW + nYZ \quad \text{(cross terms)}$$

Where:

X = L3 size, **Y** = L3 lat, **Z** = Mem Lat, **W** = FSB BW

JMP solves for coeff's (*a .. n*) that minimize the error over the input space (stepwise regression over 192 points for each trace)

Scaled Estimates by Factor

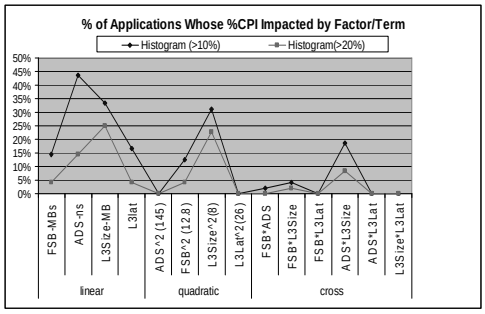
Scaled coeff estimates are the components of CPI for each term relative to the 'middle value' of the estimate

These represent the potential CPI swing for each factor -- ***very roughly***

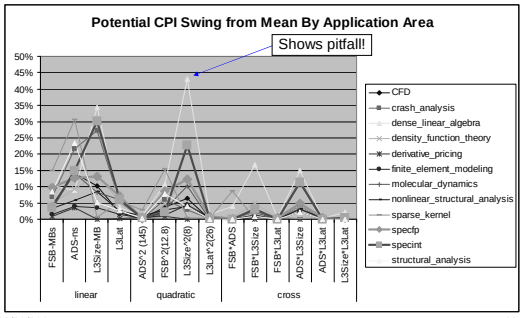
Error for stepwise regression is usually less than 2 or 3% vs measured

JMP has more accurate approaches to approximation, but do not yield functions that provide intuition

Scaled Estimates by Factor Summary



Average of Classes of Applications



Scaled Estimates by Factor

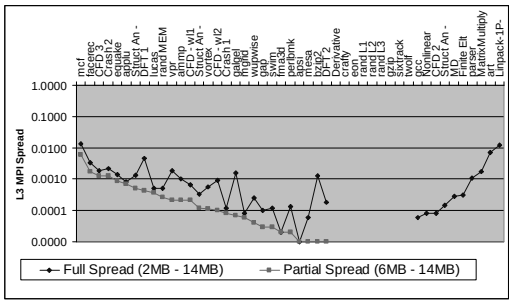
Pitfalls

- Using too-large a range of factors exaggerates impact – esp true for 'resources' such as FSB and L3 cache (crossing a knee)

Notes

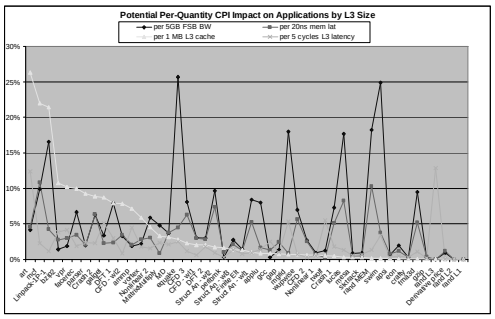
- big *linear* impact of ADS (memory latency) and L3 size, but *little or no* quadratic component
- big *quadratic* component for FSB and L3 size
- Significant cross comp. indicates that change in one factor change the curve shape of another (next slide)
- If one were trying to examine a family of alternatives, you would NOT want to simulate such wide variation in parameters

How to Know Appropriate Spread?



03/20/2005 Applications on the right – used too wide a spread! 31

Scaled Estimates by Factor



03/20/2005 Impact on CPI – each line is a uarch feature – look for peaks 32

Scaled Estimates by Factor

IPF Conclusions

- Memory latency provides the largest impact across the broadest products
  - Assuming a minimum cache size is present
- Interaction between L3 size and memory latency is a critical interaction
- Implies that data blocking can have a huge impact on these results, esp. for In-order CPU
- Can use this technique at many granularities (trace, application, app area, average)

03/20/2005 33

Future Directions

Performance projection

Projection/correlation vs other apps

Data mining to identify non-obvious critical processor features

03/20/2005 34

Conclusions

It is possible to do a very large number of applications semi-automatically

Can easily identify knees in factors

For production work, must avoid knees to make individual trade-offs

Confirms IPF need for minimal LLC size (between 2 and 6 MB per thread/core)

Confirms that despite large caches on IPF parts, memory latency has most potential performance upside for HPC apps

03/20/2005 35

Acknowledgements

Simulation: Jeff Baxter, Andrew Sun, Brian Morris, Jeff Chamberlain

HPC Applications: Mike Greenfield, Roger Golliver

PIN/Traces: Harish Patil, Robert Cohn

03/20/2005 36